QMCube (QM^3): An all-purpose suite for multiscale QM/MM calculations

Sergio Martí

Departament de Química Física i Analítica, Universitat Jaume I, 12071 Castellón, Spain

Abstract

QMCube (QM^3) is a suite written in the Python programming language, initially focused on multiscale QM/MM simulations of biological systems, but open enough to address other kinds of problems. It allows the user to combine highly efficient QM and MM programs, providing unified access to a wide range of computational methods. The suite also supplies additional modules with extra functionalities. These modules facilitate common tasks such as performing the setup of the models or process the data generated during the simulations. The design of QM^3 has been carried out considering the least number of external dependencies (only an algebra library, already included in the distribution), which makes it extremely portable. Also, the modular structure of the suite should help to expand and develop new computational methods.

1. Introduction

Multiscale Quantum mechanics / molecular mechanics (QM/MM) calculations represent, nowadays, one of the most reliable approximations to study chemical reactions in condensed media, such as aqueous and enzymatic processes. In this type of calculations, the model is divided, in the simplest approach, into a QM region comprising those atoms involved in the chemical step and, in some cases, part of the surrounding atoms. The rest of the model is then described using a MM forcefield. The way both subsystems interact leads to different flavors of QM/MM approaches, the additive scheme being one of the most popular. In the particular case of the *electrostatic embedding*, the partial charges representing the MM atoms are incorporated into the electronic self-consistent field calculation as a mono-electronic Coulomb term, perturbating the QM wave-function and, therefore, affecting the energy and the nuclear gradients.

Nowadays, it is quite common to find that a QM package allows us to include a partial charge distribution around the QM atoms, letting to perform a perturbed calculation. Furthermore, most of them enable to recover the corresponding forces acting on the partial charges. Whenever these forces are not directly provided, it could be still possible to calculate them from the electric field exercised by the QM subsystem at the partial charge location. In theory, this allows readily combining different QM and MM packages into a single calculation. This is usually unhandy in practice, requiring the development of specific in-house codes for a particular combination. This situation can be more challenging when a particular method or algorithm is not available neither in the QM package nor in the MM one. In these cases, and assuming that you have access to the source code, you have to overcome a learning curve to include the desired algorithm, which is not negligible.

Although there are other relatively recent open-source programs/suites that share similar features with QMCube, such as py-Chemshell,[1] Cuby,[2] ASE,[3] or pDynamo[4] (among others, this is not a complete list of existing codes), our suite offers a large number of interfaces with external programs (QM and MM), and provides an extremely high degree of flexibility and customization. As a result, QM^3 quickly enables the user to perform multiscale QM/MM calculations. In addition, the framework has been organized so that it can be easily used as a platform for developing new features or methodologies, although some basic programming skills are advisable.

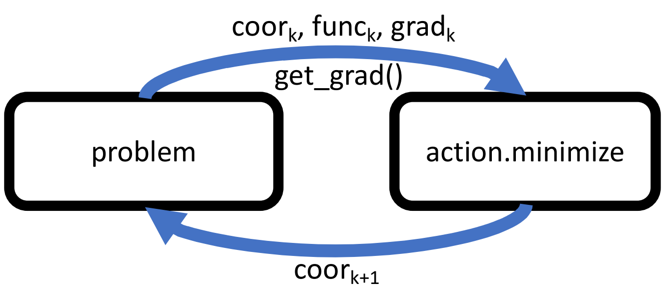
2. Code Overview

2.1 Architecture

The platform has been developed using the Python interpreted language, which facilitates the creation of the different scripts. Python is very flexible and is widely used in the scientific community. This means that the platform can be quickly integrated with other external programs, especially when they are written in this language. Besides, those procedures in the framework which are more CPU time consuming have been written in C to accelerate the calculations. In any case, the time spent by the processes of the platform will be generally smaller than the corresponding one from the execution of the external QM programs. For the shake of simplicity, the number of external dependencies has been reduced to the maximum: it only needs a mere Python installation (version 2.6 and higher), and an external algebra library such as LAPACK[5] (included in the distribution, but optimized versions like Intel-MKL,[6] macOS-Accelerate or OpenBLAS[7] can also be used). The resulting code is very portable and can be executed in almost any UNIX version (Linux, macOS, ...).

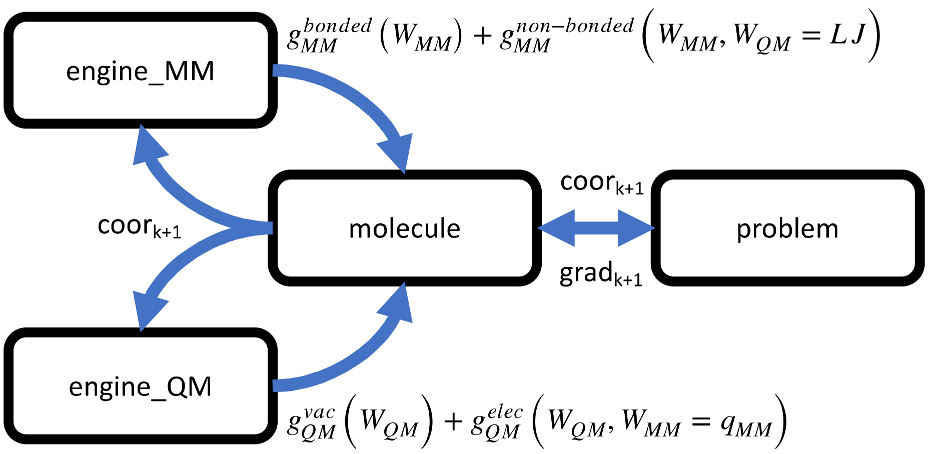
The modules of the library can be organized into three groups: i) Modules that perform ***actions***, such as molecular dynamics, geometry optimization (first and second-order algorithms), or free energy methods. ii) Modules, called ***engines***, which are devoted to interface external programs of different nature or, for instance, providing harmonic restraints. And finally, iii) those providing different varieties of tools (gridding, interpolation, setup of the models, …). Also, there is an object for handling molecular systems (***molecule***), providing general purpose methods such as selections, transformations (rotations, pruning, …), or basic IO for common formats (PDB, XYZ, Z-Matrix). Lastly, a generic ***problem*** object is introduced to assemble all the needed components (*molecule* and *engines*) in order to perform a given *action*. On one side, this approach suffers from an extra information conversion (such as coordinates or gradients) to and from the engines, but on the other hand, it introduces a large inter-operatively and universality in the tool.

As a proof of concept, let's discuss the minimization of a water dimer, one of the water molecules being described using QM (W_{QM}) and the other one through MM (W_{MM}). First, a *problem* object must be defined, which will provide the properties: *problem.coor* (six-element vector), *problem.func* (scalar), *problem.grad* (six-element vector), and the *problem.get_grad* method. Then, the minimization algorithm accesses in each iteration to the previous properties after calling the *get_grad* method, and updates the *problem.coor* with a new geometry, hopefully, closer to the minimum (see Scheme 1):



Scheme 1. Interaction between the *problem* object and the *actions.minimize* algorithm.

To be useful, the *problem* object must contain a *molecule* object and multiple engines acting on it. The *molecule* has to consider all the atoms of the model in the form of coordinates (*molecule.coor*) as well as other information (such as atom labels, atom types, partial charges). In general, the coordinates of the problem will be the same (*problem.coor* = *molecule.coor*) or a reduced subset of the molecule coordinates (as in the case of having frozen atoms in the model). In our toy model, both vectors should be the same. Finally, we need two engines, one for each Hamiltonian level: *engine_MM* and *engine_QM*. Both engines provide a *get_grad* method that acts on a molecule object used as a parameter, which accumulates the corresponding potential energy (QM or MM) and gradient vector in the molecule object (see Scheme 2):


$g_{MM}^{bonded}\left(W_{MM}\right) + g_{MM}^{non-bonded}\left(W_{MM}, W_{QM} = LJ\right)$
engine_MM
$coor_{k+1}$
molecule
$coor_{k+1}$
problem
$grad_{k+1}$
engine_QM
$g_{QM}^{vac}\left(W_{QM}\right) + g_{QM}^{elec}\left(W_{QM}, W_{MM} = q_{MM}\right)$

Scheme 2. Interaction between the *molecule* object and both QM and MM *engines*.

As a general fact, the *engine_MM* considers all the atoms in the molecule, keeping the QM atoms frozen and reduced to Lennard-Jones spheres (the charges on the QM atoms in the MM forcefield must be made zero previously to any calculation). Therefore, the gradient vector accumulated by these MM engines on the *molecule.grad* only affects to the mobile MM atoms of the system. On the other hand, the *engine_QM* is typically applied on a reduced set of atoms (substrate and, maybe, part of the surrounding amino-acids or a few water molecules). The wave function associated with the electrons of the QM atoms is then perturbed by the point charges representing the surrounding MM atoms. The gradient contribution arising from this electrostatic QM-MM interaction is then accumulated in both QM and MM atoms gradient (*molecule.grad*).
The only energy term missing after applying the different engines on the *molecule.func* and *molecule.grad* would be the gradient contribution of the Lennar-Jones term to the QM atoms. For this purpose, an additional engine (*engine_QMLJ*) must also be incorporated as part of the problem object. In short, the *problem.get_grad* method must be built using sequential calls to *engine_MM.get_grad*, *engine_QM.get_grad* and *engine_QMLJ.get_grad*. The resulting *molecule.grad* vector is then suitable to be used as a *problem.grad* (or just a portion) and then propagated to the *actions.minimize* algorithm.
In practice, the fully functional Python script needed to perform the calculation (see Listing 1) can be split into three main sections: i) import modules, ii) define the problem, and iii) apply a minimization action on the problem. This scheme reduces the number of resources consumed during the calculation since it loads only the required modules into memory. This is accomplished in lines 2-9 of the following listing:

Listing 1. Python script for optimizing a QM/MM water dimer

```
1     #!/usr/bin/env python
2     import os
3     import time
4     import qm3.mol
5     import qm3.problem
6     import qm3.engines.namd
```

```
7   import qm3.engines.dftb
8   import qm3.engines.mmint
9   import qm3.actions.minimize
10
11  os.environ["QM3_LIBDFTB"] = "./libdftb+.so"
12
13  class my_problem( qm3.problem.template ):
14      def __init__( self ):
15          qm3.problem.template.__init__( self )
16
17          self.mol = qm3.mol.molecule( "pdb" )
18          self.mol.psf_read( "psf" )
19          self.mol.nbnd_read( "non_bonded" )
20
21          os.system( "rm -vf namd.*; mkfifo namd.pipe" )
22          os.system( "NAMD_SHM=1 ./namd2 +ppn 1 inamd > namd.out &" )
23          while( not os.path.isfile( "namd.shmid" ) ):
24              time.sleep( 1 )
25          time.sleep( 2 )
26          self.emm = qm3.engines.namd.namd_shm()
27
28          self.eqm = qm3.engines.dftb.dl_dftb( self.mol, "idftb", [ 0, 1, 2 ], [ 3, 4, 5 ] )
29
30          self.fix = qm3.engines.mmint.QMLJ( self.mol, [ 0, 1, 2 ], [ 3, 4, 5 ], [] )
31
32          self.size = 3 * self.mol.natm
33          self.coor = self.mol.coor
34          self.func = 0
35          self.grad = []
36
37      def get_grad( self ):
38          self.mol.func = 0.0
39          self.mol.grad = [ 0.0 for i in range( 3 * self.mol.natm ) ]
40          self.emm.get_grad( self.mol )
41          self.eqm.get_grad( self.mol )
42          self.fix.get_grad( self.mol )
43          self.func = self.mol.func
44          self.grad = self.mol.grad[:]
45
46
47  obj = my_problem()
48  qm3.actions.minimize.fire( obj )
```

Lines 13-44 account for the whole problem definition, whose declaration derives from the **qm3.problem.template** (lines 13-15). The latter provides basic properties and methods, such as numerical gradient and Hessian evaluations. Then, the molecule is defined as a property of the problem in lines 17-19. Line 17 populates the molecule object from an existing PDB file ("pdb"). At line 18, the molecule object recovers the atom types, charges, and masses from a X-PLOR[8] compatible PSF file ("psf"). Finally, Lennard-Jones parameters for the given types are loaded from "non_bonded" file (line 19, epsilon in kcal/mol and Rmin/2 in Å).
The following blocks correspond to the inclusion of the different engines within the problem. In lines 21-26, a background process of Namd,[9] the MM engine is launched and kept reading from a UNIX pipe. This methodology is the optimal approach for external MM engines since the time spent by these kinds of programs to configure the system is usually large. Therefore, multiple calls would reduce calculation performance (although this approach is still available within QM^3). As will be discussed in the next section, we are using a patched version of Namd (**qm3.engines.namd.namd_shm**), which allows exchanging information with QM^3 using shared memory, thus increasing the speed of calculations.
The QM engine is configured on line 28. The semi-empirical DFTB+[10] program has been chosen for describing the QM water molecule at the DFTB3 level, using the 3ob-3-1 parameters.[11] The engine needs to know which atoms will be treated QM (0, 1, and 2, as C-indexing) and which ones will be surrounding point charges (MM atoms: 3, 4, and 5). In this particular case, the program has been used as a dynamic library (**qm3.engines.dftb.dl_dftb**), which again avoids using files to exchange information with QM^3. At line 30, the last engine provides the Lennard-Jones gradient for the QM atoms (**qm3.engines.mmint.QMLJ**), which is not evaluated by Namd, since the MM engine keeps frozen the QM atoms.
The ending of the problem description arises in lines 32-35, where the system size (as the total amount of variables), the coordinates, the function, and the gradient vector are declared; just the *get_grad* method remains to be defined, as shown in lines 37-44. First, the function and gradient vector of the molecule are set to zero (lines 38 and 39). Next, the gradient for the current molecular geometry is evaluated by each engine (lines 40, 41, and 42). Finally, the potential energy and the gradient vector are recovered from the molecule and mapped into the corresponding problem properties (lines 43 and 44). At this point, we only need to create a problem object (line 47) and use it as an argument to perform optimization using the FIRE[12,13] algorithm (line 48).

2.2 Engines

The engines interfaced by QM^3 can be briefly classified into three main categories: MM engines, QM engines (including those plain semi-empirical ones), and collective variables. Additionally, QM/MM able engines can also be directly used (such as CHARMM,[14] fDynamo,[15,16] or Sander[17,18]) for those cases where a particular action or algorithm is not natively available. The available engines, at present, are summarized in Table 1.

Table 1. Summary of the available QM^3 engines

Code	URL	Interface	Ref.
mol_mech	Part of QM^3	Python	
CHARMM	https://www.charmm.org/charmm/	Pipe (SHM)	14
fDynamo	https://sites.google.com/site/pdynamomodeling/	Pipe / Library	15,16
Namd	http://www.ks.uiuc.edu/Research/namd/	Pipe (SHM)	9
Amber (Sander)	http://ambermd.org	Files / Library	17,18
Lammps	https://lammps.sandia.gov/	Pipe / Python	19,20
DFT-D3	https://www.chemie.uni-bonn.de/pctc/mulliken-center/software/dft-d3/	Files / Library	21
DFTB+	https://www.dftbplus.org	Files / Library	10
Amber (SQM)		Files / Library	17,18
xTB	https://github.com/grimme-lab/xtb	Files / Library	22
deMon	http://www.demon-software.com/public_html/index.html	Files	23
Games-US	https://www.msg.chem.iastate.edu/gamess/	Files	24,25
Gaussian	http://gaussian.com/	Files	26
LSDalton	https://daltonprogram.org/	Files	27
NWChem	http://www.nwchem-sw.org/index.php/Main_Page	Files	28
Orca	https://orcaforum.kofo.mpg.de/app.php/portal	Files	29,30
Psi4	http://www.psicode.org/	Python	31
Q-Chem	http://www.q-chem.com/	Files	32
TeraChem	http://www.petachem.com/index.html	Files / MPI	33,34
BAGEL	https://nubakery.org/	Files	35
mmres	Part of QM^3	Python	36
_colvar_v	Part of QM^3	Python	37
colvar_s	Part of QM^3	Python	38–40
PLUMED	https://www.plumed.org/	Library	41,42
mmint	Part of QM^3	Python	15

The first six rows of Table 1 correspond to MM or QM/MM programs, presenting different flavors of interfaces with QM^3. Since these programs usually spend more time on initialization tasks, such as system definition or setting up non-bonding interactions, a single execution during all the calculations is the preferred option (see "Pipe" in the third column of Table 1). In this way, the external executable is fed through a UNIX pipe, asking for a given property on demand (such as energy or gradient vector). When this option is not currently available, the program is executed each time a property is demanded (referred to as "Files"). Likewise, reading the output generated by the external program can be carried out either by parsing the generated files or directly by using shared memory (indicated as "(SHM)"). We provide the needed patches along QM^3 to enable SHM based IO for some of the programs whenever the source code is available. This option enhances the communication performance, reducing global execution times (alternatively, standard calculations can be performed within the */dev/shm* folder on Linux machines). Moreover, some of the engines allow to be loaded from Python as a dynamic library (see "Library" in the third column of Table 1) or provide direct Python bindings (shown as "Python"). Finally, the internal module *mol_mech* provides a rudimentary force field intended to work with small molecules.

The next fourteen rows of Table 1 correspond to electronic QM programs, being the first four ones based on semi-empirical methods. All these engines currently interfaced provide a wide range of *ab initio*, density functional, or post Hartree-Fock methods to perform QM/MM calculations or externally enrich any of the programs with a missing method or algorithm. As stated in the introduction, most electronic structure programs allow nowadays to include point charges, representing the MM atoms, around the QM ones, thus performing a perturbed calculation. Furthermore, the majority of them let recovering the electronic gradient counterpart on the MM charges. This contribution is still feasible to be obtained via the electric field generated by the QM system on the MM charges positions or, as a last resort, classically from a fitted charge distribution of the QM atoms. The typical way to access QM engines is by means of multiple executions, using standard IO files; this is because the time spent on a QM calculation usually exceeds the time involved in setting up the input and parsing the results. The only exception is the semi-empirical based methods, where the corresponding dynamic libraries have been adapted or developed to boost performance. As a general rule, the input of all the QM engines is based on templates, thus being the calculations fully customizable and reducing the number of options to consider when developing or extending an interface.

The last five rows of Table 1 correspond to modules designed to add soft restraints (*mmres*), to make use of collective variables (*_colvar_v*, *colvar_s*, and *PLUMED*), and introducing additional interactions between QM and MM kind engines (*mmint*). In the case of the collective variables, QM^3 allows applying harmonic restraints on both the electrostatic potential generated by a classical environment on a given atom or linear combination,[37] as well as on the s path-based collective variable[38–40] to conduct potential of mean force (PMF) calculations. Besides, the interface with PLUMED[41,42] provides access to a wider range of collective variables or even enables metadynamics.[43,44]

Finally, provided that there are covalent boundaries between QM and MM regions, the link atom approach[45] has been adopted within QM^3. The main reason is that this convention allows using virtually any QM engine without introducing any modification in the code (whenever it is available). Since the responsibility for fixing the boundary between the QM and MM engines lies on the user, we help generate the necessary amount of additional soft restraints in the form of bonds, angles, and dihedrals. The automatic code generated for the hydrogen peroxide is shown in the following listing, where half of the molecule is considered QM (atoms 0 and 1, C-indexing), and the rest is MM (atoms 2 and 3). The user still must provide appropriate values of force constants and reference parameters for the restraints. Notice that the corresponding energies are scaled by zero ("*ffac*", thus not taken into account) since the MM engine already considers them, and only the gradient contribution on the QM atom is needed ("*gfac*").

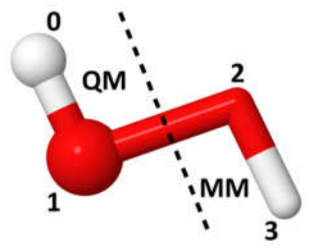
Listing 2. Example of QM/MM partition using link atoms in the hydrogen peroxide molecule.

```
self.exc = []

#----------------------------------------------------------------
self.exc.append( qm3.engines.mmres.distance( kumb_kJ/mol.A^2, xref_A, [ 1, 2 ] ) )
self.exc[-1].ffac = 0.0
self.exc[-1].gfac = [ 1.0, 0.0 ]

#----------------------------------------------------------------
self.exc.append( qm3.engines.mmres.angle( kumb_kJ/mol.rad^2, xref_deg, [ 1, 2, 3 ] ) )
self.exc[-1].ffac = 0.0
self.exc[-1].gfac = [ 1.0, 0.0, 0.0 ]

#----------------------------------------------------------------
self.exc.append( qm3.engines.mmres.dihedral( { n: [ frc_kJ/mol, dsp_deg ] }, [ 0, 1, 2, 3 ] ) )
self.exc[-1].ffac = 0.0
self.exc[-1].gfac = [ 1.0, 1.0, 0.0, 0.0 ]
```



2.3 Actions

Once the problem object has been defined, containing a molecule object and the appropriate number of engines, the next step is to perform a particular action. Currently, there are several actions available in QM^3, listed in the Table 2:

Table 2. Summary of the available QM^3 actions

Action (module)	Method / Algorithm	Ref.
minimize	FIRE	[12,13]
	Conjugate Gradient PLUS	[46]
	Partitioned Rational Function Optimization (P-RFO)	[47]
	Baker's mode-following	[48–50]
genetic	Differential Evolution	[51]
paths	Local Quadratic Approximation (LQA)	[52,53]
dynamics	Velocity-Verlet	[54]
	Langevin-Verlet	[55,56]
string	On-the-fly String	[40,57,58]
grote_hynes	Transmission Coefficients	[59–61]
neb	Nudged Elastic Band	[62–64]

The most common action corresponds to the optimization of the geometrical coordinates of the model; this is achieved through different algorithms implemented in the *minimize* module, depending on the information used during the optimization: FIRE or the Conjugate Gradient when using the gradient vector and the rational function optimization or the Baker algorithm whenever the Hessian matrix is available (second-order algorithms). In the two later, the mode being followed during the optimization can be initially selected and made it change during the process by means of a vector overlap mechanism. Indeed, the quadratic methods can also locate transition structures (TS) using a micro/macro iterative scheme.[65] In such a case, they would be applied to a small subset of atoms comprising mostly the reacting fragments/molecules (or core), while the rest of the system (or environment) is kept fully relaxed at each core iteration through the gradient vector evaluation. Once a TS has been located and properly characterized by inspecting Hessian matrix eigenvalues, the corresponding reactions paths towards reactants and products can be traced down based on the local quadratic approximation as implemented in the *paths* module. This module can also use a micro/macro scheme and apply special conditions (such as initial longer/shorter steps or trying to avoid, in a rudimentary way, recrossings). Likewise, it allows performing a restart of the algorithm for those cases where the number of steps is not enough or a crash has occurred. An alternative way to obtain the minimum energy path (MEP) is through the nudged elastic band method (NEB). Our NEB implementation draws on the most appropriate method for the tangent evaluation at each node,[63] based on the potential energy. In addition to the serialized version, it can also be run in parallel through an MPI interface, being able to use chunks if the number of nodes of the band does not fit the number of available processors. Finally, a zero-order genetic algorithm is also provided for performing optimizations on systems with a large number of local minima (not necessarily chemical models). In particular, three different flavors of the differential evolution are available depending on the implementation: a serialized version (based on "*rand/1*"), an MPI version, and a threaded one (both based on "*rand-to-best/1*"), all of them using binomial crossovers. Although the parallel versions are generally the fastest, the serial one allows, by design, a larger number of crossings among the generated populations.
Another common action is performing molecular dynamics (MD) simulations. QM^3 incorporates the Velocity-Verlet integrator in two different ensembles: NVE and NVT. While no particular action is required to constraint the internal energy, the Langevin thermostat is the preferred way to keep the average temperature constant. The combination of the Berendsen thermostat and barostat (NpT) was explored, using the numerical derivative of the internal energy with respect to the volume for calculating the instantaneous pressure. However, it was dismissed since some MM programs do not allow changing the system cell dimensions during the simulations. Anyhow, MD simulations in the NVT ensemble allow to estimate free energies using different methodologies, such as the PMF through the combination of the umbrella sampling (US) and the weighted histogram analysis method[66,67] (WHAM), or via the free energy perturbation method (FEP) along the reaction path.[68,69] An alternative way to obtain free energy surfaces (FES) is by combining MD with the on-the-fly string method, where the free energy is recovered from the mean forces acting on the evenly spaced nodes of a string defined by a single collective variable. Once a string has converged, all the information generated, as the node positions and their corresponding averaged metrics (associated with the usual curvilinear

nature of the geometrical coordinates defining the collective variable), can also be reused to properly define a collective path variable, such as the s-coordinate (*colvar_s*), and perform a standard 1D-PMF with it.
Finally, MD in the NVE ensemble can be useful to evaluate transmission coefficients using both the flux theory[70] or the Grote-Hynes approach[59–61] (GH). While the former does not require any special treatment beyond NVE-MD or the projection of the reaction coordinate components of the initial velocities, the *grote_hynes* module allows, by means of Lagrange multipliers, to remove and record the forces acting on the reaction coordinate during the MD on the transition structure, and thus to estimate the friction kernel from their autocorrelation.

2.4 Utilities

In addition to the problem object building blocks, different tools have been incorporated, most of them fulfilling internal dependencies but which enrich the QM^3 suite providing more functionalities. Some of them are listed in Table 3 and presented below.

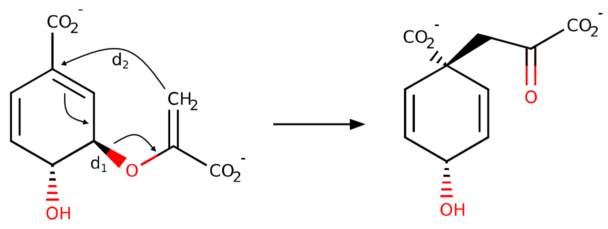
Table 3. Some of the available QM^3 utilities

Module	Description
interpolation	Different flavors of interpolation methods
grids	Tools for working with grids
stats	Basic statistics and clustering methods
utils	Geometry, Normal modes, Hessian updaters
mpi / msi	MPI Python bindings and in-house based ones (msi)
shm	Shared Memory Python bindings
free_energy	Tools for estimating free energies
prepare	Methods for systems setup

The *interpolation* module allows different types of mono-dimensional interpolations, which can also be used for two-dimensional interpolations on regular data (performing orthogonal interpolations). Each interpolator is coded as an object, implementing a "*calc*" method, which returns the interpolated value and the corresponding first derivative. Part of the interpolating alternatives included rely on Hermite-based splines functions, such as those of Steffen,[71] Akima,[72] or Fritsch-Carlson,[73] but also based on Lagrange or Gaussian methods, the latter being able to smooth the data.
Another particularly handy module intended for PESs or FESs processing is the *grid* module. It supports to parse unsorted data (changing in x or y-axis) with any set of columns and transforms the data into a smooth regular grid by means of Gaussian interpolation. Moreover, this module can directly generate 3D-plots when the matplotlib library[74] is accessible.
The *stats* and *utils* modules introduce an extra set of available tools. While the first one makes simple to cluster sparse data using the K-means++ algorithm,[75] the second contains a mixture of broad range functions, such as basic geometrical coordinates, superimposing of cartesian sets,[71–73] vibrational normal modes analysis,[15] or Hessian matrix updating techniques.[79]
The *free_energy* module provides different procedures for integrating free energy differences, depending on the particular methodology applied during the simulations. For instance, the integration of FEP calculations is achieved using a derived version of the Bennet acceptance ratio[80,81] (BAR), while the associated error is estimated using the first-order expansion.[68] On the other hand, QM^3 offers two different approaches when the US technique is used: the WHAM method[15,67] for rendering 1D-PMFs, and the Umbrella Integration (UI) for both 1D[82,83] and 2D-PMFs.[84]
Another general module is *prepare*, which aims to facilitate some aspects of the initial configuration of the model. It allows counterions to be added around a protein or solute, placing them in electrostatically optimal positions in a similar way to the "Sodium" [85] program: a grid is constructed around the protein, and the electrostatic potential is calculated at each point, determining the best location for the counterion. This procedure is repeated for each ion under some restrictions, such as the ion-protein or ion-ion distances. The resulting neutrally charged model can be then solvated using different types of boxes, using the external program Packmol[86] to generate them. Although all of these functions are available in other calculation suites, they complement QM^3 as they help to perform the model setup within a common workflow.
The three remaining modules have been designed to facilitate communication among parallel instances of QM^3 tasks or external engines. In the first case, a reduced set of MPI instructions have been wrapped into a Python binary module, affording plain IO and synchronization in parallel MPI runs. An additional module called *msi* (standing for message socket interface), based on both *UNIX* and *internet* sockets, is also provided for those cases where MPI is not available. The remaining *shm* module is also a Python binary wrapper, designed for working with shared memory to speed up IO against patched external engines, thus avoiding the use of standard files for communication.

3. Applications

In this section, we present a QM/MM enzymatic reaction case study. The selected process is the conversion of chorismate to prephenate, for which the mechanism is well established since it has been extensively visited in the literature, thus being a perfect candidate for benchmarking purposes. Moreover, it is a concerted unimolecular transformation, which allows a clean partition between the QM and MM regions of the model (without the requirement of any QM-MM link atom treatment) and reduces the number of chemical steps to a single one (see Scheme 3).



Scheme 3. Schematic representation of the reaction mechanism for the chorismate to prephenate rearrangement.

The starting point was a functional monomer with chorismate mutase activity from *Methanococcus jannaschii* (mMjCM, PDB id 2GTV).[87] The transition state analog present in the initial X-ray structure was manually modified into a chorismate molecule. Then, hydrogen atoms were added according to the pKa values rendered by the empirical PropKa-3.1 program[88] for a pH equal to 6.5. The protonation states of the different histidine residues, delta or epsilon, were assigned by visual inspection, trying to encourage at any time the presence of hydrogen bonds with the surrounding amino acid residues. A single counterion (Na^+) was needed to fulfill the model electroneutrality and was placed into an optimal electrostatic position using the *counter_ions* method of the *prepare* module. Thereupon, the system was placed in a 90 Å × 72 Å × 69 Å orthorhombic box of water molecules generated with the Packmol program through the *prepare.solvate* method (see listing L1 in the SI for these two steps). The resulting model was optimized using a QM^3 *problem* object comprising a Namd engine, which used the Charmm22 force field.[89] The force-switch scheme was adopted for the nonbonded interactions, with a cutoff radius ranging from 14.5 to 16.5 Å, combined with periodic boundary conditions. A combination of the steepest descent followed by the FIRE algorithms were applied until reaching convergence (listing L2 in the SI). Afterward, a total of 20 ns of classical MD was run with Namd in the NVT ensemble, using the Langevin-Verlet integrator with a time step of 1 fs at a temperature of 300 K.

The previous *problem* object was thereupon extended with a DFTB+/3ob-3-1 engine, which opens the gate to the electronic structure treatment and thus to study the chemical reactivity. Consequently, some additional steps were carried out (listing L3 in the SI), such as defining the QM region (the 24 atoms of the substrate) and the MM one that can interact with the former (any atom up to 20 Å from the substrate, 5324 atoms in total). From this point on, only these two sets of atoms were considered as the active part of the *problem*, being their coordinates modified and keeping the rest of the model frozen.

Once the model was set up, relaxed, and with all its parts properly defined, the next step was to explore the PES for the conversion of chorismate into prephenate. For this purpose, a NEB was performed to characterize the energetics of the process, followed by the vibrational analysis of the located stationary structures. Initially, the band endpoints, corresponding to reactants and products, were obtained through a global QM/MM optimization of the active atoms (listing L4 in the SI). A total of 40 intermediate nodes were generated by linear interpolation, and the resulting band was minimized using the parallel version of the algorithm implemented in QM^3 (listing L5 in the SI). The MM atoms were kept fully optimized at each NEB step to ensure that any substrate change was accompanied by the environment, as in the micro/macro iterative scheme presented above (see section 2.3). The predicted energy profile shows an activation energy barrier of 15 kcal/mol and an exothermic process by -7 kcal/mol, as depicted in Figure 1a.

The analysis of the normal modes (listing L6 in the SI) confirms the presence of a transition structure (see Figure 1b) with a single negative eigenvalue of 307 cm^{-1}, providing a valid starting point for a later transition structure refinement.

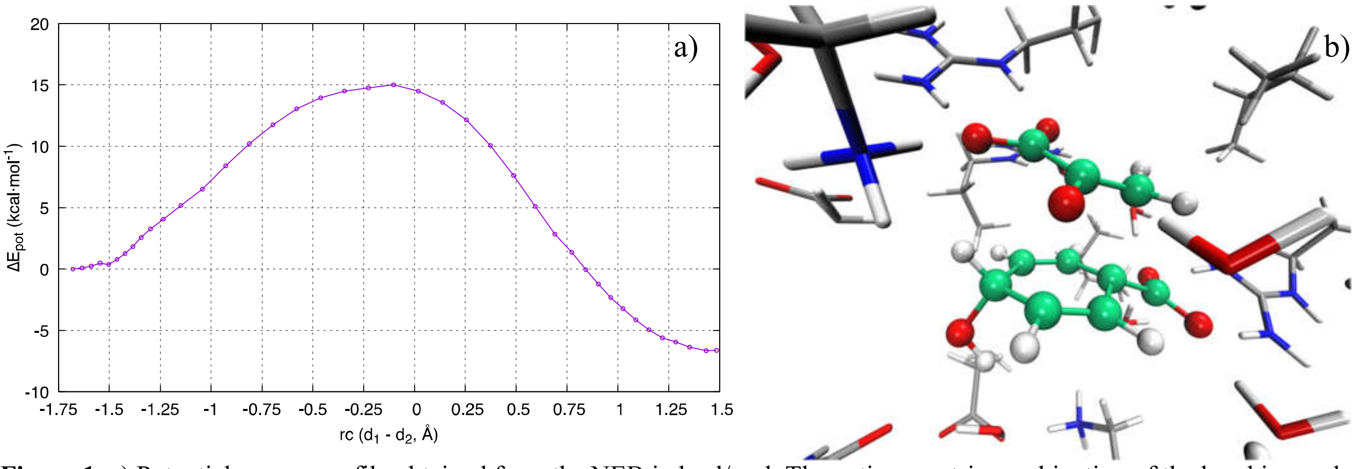


Figure 1. a) Potential energy profile obtained from the NEB in kcal/mol. The antisymmetric combination of the breaking and forming bonds (d_1 - d_2, see Scheme 3) has been mapped in the abscissa. b) Detail of the active site for the node with the highest energy, assigned to the transition structure (QM substrate depicted in CPK).

The chain of nodes obtained via NEB is an ideal starting point for exploring the minimum free energy path (MFEP), using techniques as the on-the-fly string method. In this case, the collective variable was built using the forming and breaking bonds (d_2 and d_1 in scheme 3), and the total number of nodes was set to 54. Due to the requirement of the string reparameterization, all nodes must run at the same time; this can be achieved sequentially (iterating over each node at a time) or in parallel, using as many processes as the string nodes or using chunks (i.e., iterating a small number of nodes on each process). The latter approach has been adopted using two servers executing nine processes each and, consequently, chunks of three nodes per process (see listing L7 in the SI). A total of 5 ps of MD were run in the NVT ensemble for each string node (500 steps for equilibration and 4500 for acquisition, with a time

step of 1 fs). The integration of the average projected forces acting on each node leads to an estimation of the activation free energy for the conversion of 13.4 kcal/mol, with an exergonic product at -11.6 kcal/mol, as observed in Figure 2.
Since the reaction can be appropriately described using the antisymmetric combination of breaking-forming bonds, a regular mono-dimensional PMF can be readily carried out, based on the combination of the US and WHAM methodologies. For this purpose, a total of 54 windows were run, allowing to sample the interval [-1.70:1.64] Å of the reaction coordinate ($\Delta\xi$ = 0.063 Å). As in the previous calculations, the NVT ensemble with a time step of 1 fs was selected, and the same number of MD steps were performed (0.5 ps for relaxation and 4.5 ps for acquisition, the endpoint of the string calculation being the starting geometry of each window). The force constant of the harmonic potential applied to restraint the coordinate at each window was 669 kcal·mol^{-1}·Å^{-2}, a value stiff enough to control the reaction coordinate and ensure the overlap between neighboring windows. Given no information exchange among the different windows is needed, all calculations were distributed in the available processors for running in parallel (see listing L8 of the SI). The application of the WHAM methodology renders an activation free energy of 14.5 kcal/mol, with a slightly earlier transition structure located at a value of -0.27 Å of the reaction coordinate. Regarding the formation of prephenate, the obtained PMF shows that it is an exergonic process of -11.7 kcal/mol (see Figure 2).
The last methodology used to estimate the proposed reaction mechanism energetics is also a PMF calculation, but using a path-based collective variable as a reaction coordinate. The main advantage of this approach is reducing the PMF dimensionality, whatever the number of coordinates involved in the reaction process, just focusing on the reaction tube. For this purpose, additional information should be provided, such as the distance metric tensor, which depends on the collective variable coordinates. Fortunately, this can be obtained from the previous string calculation and the starting points for each window. The s coordinate ranged from zero to 6.3 for the 54 windows, and the force constant used for the umbrellas was 478 kcal/mol. As in the previous calculations, 5ps NVT molecular dynamics were run for each window (0.5 plus 4.5 ps) with an integration step of 1 fs, being all the windows submitted in parallel. The umbrella integration methodology on the data produced at each window leads to an activation free energy of 15.1 kcal/mol and an exergonic product of -9.4 kcal/mol, as shown in Figure 2. The difference of the activation free energy, compared with the one obtained with the previous PMF, could correspond to the improvement of the reaction coordinate obtained with the string method, in addition to the statistical error. For comparing both profiles, the s reaction coordinate has been mapped over the antisymmetric combination, using the average value drained from each window by means of the recovered density (see listings L9 and L10 of the SI).

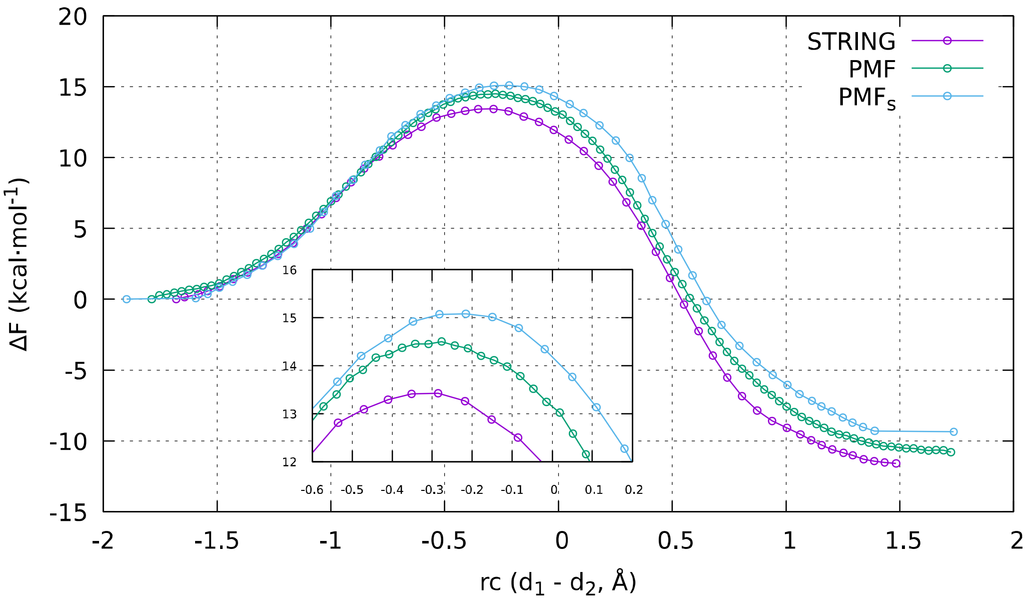


Figure 2. Free energy profiles obtained applying different techniques: on-the-fly string method (STRING), standard potential of mean force (PMF), and path-based PMF (PMF$_S$).

Just a note of caution: the reader must bear in mind that this section is merely demonstrative; thus, the time lengths used to carry out the different MD should generally be longer to obtain robust conclusions and minimize statistical errors. All in all, the computational results are close to the experimentally derived value (kcat = 3.2 s^{-1} at 303 K,[87] thus an approximate value for $\Delta G^{\ddagger}$ of ca 17 kcal·mol^{-1}).
Finally, for illustrative purposes, an additional example of an on-the-fly FEP calculation for a gas-phase molecule in two electronic spin states has also been included in the supporting information.

4. Conclusions

The QM^3 suite has been developed as a calculation platform to perform QM/MM simulations on biological processes, such as studying enzymatic catalysis, and we believe it can be useful for developing new computational methods. The frame abstraction layer provided by its modular and object-oriented structure makes it easy to adapt and expand. In addition, the use of template-based QM engines facilitates the development of new interfaces. Finally, the suite contains all the necessary ingredients to i) set up the

computational models, ii) perform different kinds of calculations, and iii) process and analyze the generated results, all in a single environment.
QM^3 began as a particular solution to our research needs (not having to rewrite the same code when changing from one program to another), but we think it is mature enough to give it a broad and generic use.
The QM^3 is freely available via a public GitHub repository (https://github.com/sergio-marti/qm3.git).

Associated Content: Supporting Information
Listings of the different fully-functional Python scripts (L1-L10) used in the study of the monomeric chorismate mutase from *Methanococcus jannaschii* (mMjCM). An additional two electronic spin states FEP calculation is also included.

Author Information
Corresponding Author
*E-mail: smarti@uji.es

ORCID
Sergio Martí: 0000-0002-1087-7143

Funding
This work was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades (Grant PGC2018-094852-B-C21), Universitat Jaume I (Project No. UJI-B2017-31), and Generalitat Valenciana (Grant AICO/2019/195). The author acknowledges computational resources from the Servei d'Informàtica of Universitat Jaume I.

Acknowledgments
The author is grateful to V. Moliner and I. Tuñón for patiently reading the manuscript and providing valuable comments.

Data Availability Statement
The data that support the findings of this study are available from the corresponding author upon reasonable request.

Notes
The author declares no competing financial interest.

References
(1) Lu, Y.; Farrow, M. R.; Fayon, P.; Logsdail, A. J.; Sokol, A. A.; Catlow, C. R. A.; Sherwood, P.; Keal, T. W. Open-Source, Python-Based Redevelopment of the ChemShell Multiscale QM/MM Environment. *J. Chem. Theory Comput.* **2019**, *15* (2), 1317–1328. https://doi.org/10.1021/acs.jctc.8b01036.
(2) Řezáč, J. Cuby: An Integrative Framework for Computational Chemistry. *J. Comput. Chem.* **2016**, *37* (13), 1230–1237. https://doi.org/10.1002/jcc.24312.
(3) Hjorth Larsen, A.; JØrgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dułak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermode, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; SchiØtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The Atomic Simulation Environment - A Python Library for Working with Atoms. *Journal of Physics Condensed Matter*. Institute of Physics Publishing June 7, 2017, pp 273002–273031. https://doi.org/10.1088/1361-648X/aa680e.
(4) Field, M. J. The PDynamo Program for Molecular Simulations Using Hybrid Quantum Chemical and Molecular Mechanical Potentials. *J. Chem. Theory Comput.* **2008**, *4* (7), 1151–1161. https://doi.org/10.1021/ct800092p.
(5) Anderson, E.; Bai, Z.; Bischof, C.; Blackford, S.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; Sorensen, D. *LAPACK Users' Guide*, Third.; Society for Industrial and Applied Mathematics: Philadelphia, PA, 1999.
(6) Intel® Math Kernel Library (MKL) https://software.intel.com/en-us/mkl.
(7) Wang, Q.; Zhang, X.; Zhang, Y.; Yi, Q. AUGEM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*; ACM Press: New York, New York, USA, 2013; pp 1–12. https://doi.org/10.1145/2503210.2503219.
(8) Brünger, A. T. *X-PLOR Version 3.1: A System for X-Ray Crystallography and NMR*; Yale University Press, Ed.; 1992.
(9) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. Scalable Molecular Dynamics with NAMD. *J. Comput. Chem.* **2005**, *26* (16), 1781–1802. https://doi.org/10.1002/jcc.20289.
(10) Aradi, B.; Hourahine, B.; Frauenheim, T. DFTB+, a Sparse Matrix-Based Implementation of the DFTB Method†. **2007**. https://doi.org/10.1021/JP070186P.
(11) Gaus, M.; Goez, A.; Elstner, M. Parametrization and Benchmark of DFTB3 for Organic Molecules. *J. Chem. Theory Comput.* **2013**, *9* (1), 338–354. https://doi.org/10.1021/ct300849w.
(12) Bitzek, E.; Koskinen, P.; Gähler, F.; Moseler, M.; Gumbsch, P. Structural Relaxation Made Simple. *Phys. Rev. Lett.* **2006**, *97* (17), 170201–170205. https://doi.org/10.1103/PhysRevLett.97.170201.
(13) Guénolé, J.; Nöhring, W. G.; Vaid, A.; Houllé, F.; Xie, Z.; Prakash, A.; Bitzek, E. Assessment and Optimization of the Fast Inertial Relaxation Engine (FIRE) for Energy Minimization in Atomistic Simulations and Its Implementation in LAMMPS. *arXiv:1908.02038 [physics.comp-ph]* **2019**.

(14) Brooks, B. R.; Brooks, C. L.; Mackerell, A. D.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. CHARMM: The Biomolecular Simulation Program. *J. Comput. Chem.* **2009**, *30* (10), 1545–1614. https://doi.org/10.1002/jcc.21287.

(15) Field, M. J. *A Practical Introduction to the Simulation of Molecular Systems*; Cambridge University Press, 1999.

(16) Field, M. J.; Albe, M.; Bret, C.; Proust-De Martin, F.; Thomas, A. The Dynamo Library for Molecular Simulations Using Hybrid Quantum Mechanical and Molecular Mechanical Potentials. *J. Comput. Chem.* **2000**, *21* (12), 1088–1100. https://doi.org/10.1002/1096-987X(200009)21:12<1088::AID-JCC5>3.0.CO;2-8.

(17) Case, D. A.; Cheatham, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J.; Wang, B.; Woods, R. J. The Amber Biomolecular Simulation Programs. *J. Comput. Chem.* **2005**, *26* (16), 1668–1688. https://doi.org/10.1002/jcc.20290.

(18) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An Overview of the Amber Biomolecular Simulation Package. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2013**, *3* (2), 198–210. https://doi.org/10.1002/wcms.1121.

(19) Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.* **1995**, *117* (1), 1–19. https://doi.org/10.1006/JCPH.1995.1039.

(20) Brown, W. M.; Kohlmeyer, A.; Plimpton, S. J.; Tharrington, A. N. Implementing Molecular Dynamics on Hybrid High Performance Computers - Particle-Particle Particle-Mesh. *Comput. Phys. Commun.* **2012**, *183* (3), 449–459. https://doi.org/10.1016/j.cpc.2011.10.012.

(21) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A Consistent and Accurate Ab Initio Parametrization of Density Functional Dispersion Correction (DFT-D) for the 94 Elements H-Pu. *J. Chem. Phys.* **2010**, *132* (15), 154104–154122. https://doi.org/10.1063/1.3382344.

(22) Bannwarth, C.; Ehlert, S.; Grimme, S. GFN2-XTB - An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions. *J. Chem. Theory Comput.* **2019**, *15* (3), 1652–1671. https://doi.org/10.1021/acs.jctc.8b01176.

(23) Koster, A. M.; Geudtner, G.; Alvarez-Ibarra, A.; Calaminici, P.; Casida, M. E.; Carmona-Espindola, J.; Dominguez, V. D.; Flores-Moreno, R.; Gamboa, G. U.; Goursot, A.; Heine, T.; Ipatov, A.; de la Lande, A.; Janetzko, F.; del Campo, J. M.; Mejia-Rodriguez, D.; Reveles, J. U.; Vasquez-Perez, J.; Vela, A.; Zuniga-Gutierrez, B.; Salahub, D. R. DeMon2k.

(24) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. General Atomic and Molecular Electronic Structure System. *J. Comput. Chem.* **1993**, *14* (11), 1347–1363. https://doi.org/10.1002/jcc.540141112.

(25) Dykstra, C. E. *Theory and Applications of Computational Chemistry : The First Forty Years*; Elsevier, 2005.

(26) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Mennucci, B.; Petersson, G. A.; Nakatsuji, H.; Caricato, M.; Li, X.; Hratchian, H. P.; Izmaylov, A. F.; Bloino, J.; Zheng, G.; Sonnenberg, J. L.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Montgomery Jr., J. A.; Peralta, J. E.; Ogliaro, F.; Bearpark, M.; Heyd, J. J.; Brothers, E.; Kudin, K. N.; Staroverov, V. N.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Rega, N.; Millam, J. M.; Klene, M.; Knox, J. E.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Zakrzewski, V. G.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Dapprich, S.; Daniels, A. D.; Farkas, Ö.; Foresman, J. B.; Ortiz, J. V; Cioslowski, J.; Fox, D. J. Gaussian09.

(27) Aidas, K.; Angeli, C.; Bak, K. L.; Bakken, V.; Bast, R.; Boman, L.; Christiansen, O.; Cimiraglia, R.; Coriani, S.; Dahle, P.; Dalskov, E. K.; Ekström, U.; Enevoldsen, T.; Eriksen, J. J.; Ettenhuber, P.; Fernández, B.; Ferrighi, L.; Fliegl, H.; Frediani, L.; Hald, K.; Halkier, A.; Hättig, C.; Heiberg, H.; Helgaker, T.; Hennum, A. C.; Hettema, H.; Hjertenaes, E.; Høst, S.; Høyvik, I.-M.; Iozzi, M. F.; Jansík, B.; Jensen, H. J. A.; Jonsson, D.; Jørgensen, P.; Kauczor, J.; Kirpekar, S.; Kjaergaard, T.; Klopper, W.; Knecht, S.; Kobayashi, R.; Koch, H.; Kongsted, J.; Krapp, A.; Kristensen, K.; Ligabue, A.; Lutnaes, O. B.; Melo, J. I.; Mikkelsen, K. V.; Myhre, R. H.; Neiss, C.; Nielsen, C. B.; Norman, P.; Olsen, J.; Olsen, J. M. H.; Osted, A.; Packer, M. J.; Pawlowski, F.; Pedersen, T. B.; Provasi, P. F.; Reine, S.; Rinkevicius, Z.; Ruden, T. A.; Ruud, K.; Rybkin, V. V.; Sałek, P.; Samson, C. C. M.; de Merás, A. S.; Saue, T.; Sauer, S. P. A.; Schimmelpfennig, B.; Sneskov, K.; Steindal, A. H.; Sylvester-Hvid, K. O.; Taylor, P. R.; Teale, A. M.; Tellgren, E. I.; Tew, D. P.; Thorvaldsen, A. J.; Thøgersen, L.; Vahtras, O.; Watson, M. A.; Wilson, D. J. D.; Ziolkowski, M.; Ågren, H. The Dalton Quantum Chemistry Program System. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2014**, *4* (3), 269–284. https://doi.org/10.1002/wcms.1172.

(28) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H. J. J.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. NWChem: A Comprehensive and Scalable Open-Source Solution for Large Scale Molecular Simulations. *Comput. Phys. Commun.* **2010**, *181* (9), 1477–1489. https://doi.org/10.1016/J.CPC.2010.04.018.

(29) Neese, F. The ORCA Program System. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012**, *2* (1), 73–78. https://doi.org/10.1002/wcms.81.

(30) Neese, F. Software Update: The ORCA Program System, Version 4.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8* (1), e1327. https://doi.org/10.1002/wcms.1327.

(31) Parrish, R. M.; Burns, L. A.; Smith, D. G. A.; Simmonett, A. C.; DePrince, A. E.; Hohenstein, E. G.; Bozkaya, U.; Sokolov, A. Y.; Di Remigio, R.; Richard, R. M.; Gonthier, J. F.; James, A. M.; McAlexander, H. R.; Kumar, A.; Saitow, M.; Wang, X.; Pritchard, B. P.; Verma, P.; Schaefer, H. F.; Patkowski, K.; King, R. A.; Valeev, E. F.; Evangelista, F. A.; Turney, J. M.; Crawford, T. D.; Sherrill, C. D. Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability. *J. Chem. Theory Comput.* **2017**, *13* (7), 3185–3197.

https://doi.org/10.1021/acs.jctc.7b00174.
(32) Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, A. T. B.; Wormit, M.; Kussmann, J.; Lange, A. W.; Behn, A.; Deng, J.; Feng, X.; Ghosh, D.; Goldey, M.; Horn, P. R.; Jacobson, L. D.; Kaliman, I.; Khaliullin, R. Z.; Kuś, T.; Landau, A.; Liu, J.; Proynov, E. I.; Rhee, Y. M.; Richard, R. M.; Rohrdanz, M. A.; Steele, R. P.; Sundstrom, E. J.; Woodcock, H. L.; Zimmerman, P. M.; Zuev, D.; Albrecht, B.; Alguire, E.; Austin, B.; Beran, G. J. O.; Bernard, Y. A.; Berquist, E.; Brandhorst, K.; Bravaya, K. B.; Brown, S. T.; Casanova, D.; Chang, C.-M.; Chen, Y.; Chien, S. H.; Closser, K. D.; Crittenden, D. L.; Diedenhofen, M.; DiStasio, R. A.; Do, H.; Dutoi, A. D.; Edgar, R. G.; Fatehi, S.; Fusti-Molnar, L.; Ghysels, A.; Golubeva-Zadorozhnaya, A.; Gomes, J.; Hanson-Heine, M. W. D.; Harbach, P. H. P.; Hauser, A. W.; Hohenstein, E. G.; Holden, Z. C.; Jagau, T.-C.; Ji, H.; Kaduk, B.; Khistyaev, K.; Kim, J.; Kim, J.; King, R. A.; Klunzinger, P.; Kosenkov, D.; Kowalczyk, T.; Krauter, C. M.; Lao, K. U.; Laurent, A. D.; Lawler, K. V.; Levchenko, S. V.; Lin, C. Y.; Liu, F.; Livshits, E.; Lochan, R. C.; Luenser, A.; Manohar, P.; Manzer, S. F.; Mao, S.-P.; Mardirossian, N.; Marenich, A. V.; Maurer, S. A.; Mayhall, N. J.; Neuscamman, E.; Oana, C. M.; Olivares-Amaya, R.; O'Neill, D. P.; Parkhill, J. A.; Perrine, T. M.; Peverati, R.; Prociuk, A.; Rehn, D. R.; Rosta, E.; Russ, N. J.; Sharada, S. M.; Sharma, S.; Small, D. W.; Sodt, A.; Stein, T.; Stück, D.; Su, Y.-C.; Thom, A. J. W.; Tsuchimochi, T.; Vanovschi, V.; Vogt, L.; Vydrov, O.; Wang, T.; Watson, M. A.; Wenzel, J.; White, A.; Williams, C. F.; Yang, J.; Yeganeh, S.; Yost, S. R.; You, Z.-Q.; Zhang, I. Y.; Zhang, X.; Zhao, Y.; Brooks, B. R.; Chan, G. K. L.; Chipman, D. M.; Cramer, C. J.; Goddard, W. A.; Gordon, M. S.; Hehre, W. J.; Klamt, A.; Schaefer, H. F.; Schmidt, M. W.; Sherrill, C. D.; Truhlar, D. G.; Warshel, A.; Xu, X.; Aspuru-Guzik, A.; Baer, R.; Bell, A. T.; Besley, N. A.; Chai, J.-D.; Dreuw, A.; Dunietz, B. D.; Furlani, T. R.; Gwaltney, S. R.; Hsu, C.-P.; Jung, Y.; Kong, J.; Lambrecht, D. S.; Liang, W.; Ochsenfeld, C.; Rassolov, V. A.; Slipchenko, L. V.; Subotnik, J. E.; Van Voorhis, T.; Herbert, J. M.; Krylov, A. I.; Gill, P. M. W.; Head-Gordon, M. Advances in Molecular Quantum Chemistry Contained in the Q-Chem 4 Program Package. *Mol. Phys.* **2015**, *113* (2), 184–215. https://doi.org/10.1080/00268976.2014.952696.
(33) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics. *J. Chem. Theory Comput.* **2009**, *5* (10), 2619–2628. https://doi.org/10.1021/ct9003004.
(34) Titov, A. V.; Ufimtsev, I. S.; Luehr, N.; Martinez, T. J. Generating Efficient Quantum Chemistry Codes for Novel Architectures. *J. Chem. Theory Comput.* **2013**, *9* (1), 213–221. https://doi.org/10.1021/ct300321a.
(35) Shiozaki, T. BAGEL: Brilliantly Advanced General Electronic-Structure Library. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8* (1), e1331. https://doi.org/10.1002/wcms.1331.
(36) Rackers, J. A.; Wang, Z.; Lu, C.; Laury, M. L.; Lagardère, L.; Schnieders, M. J.; Piquemal, J. P.; Ren, P.; Ponder, J. W. Tinker 8: Software Tools for Molecular Design. *J. Chem. Theory Comput.* **2018**, *14* (10), 5273–5289. https://doi.org/10.1021/acs.jctc.8b00529.
(37) García-Meseguer, R.; Martí, S.; Ruiz-Pernía, J. J.; Moliner, V.; Tuñón, I. Studying the Role of Protein Dynamics in an SN2 Enzyme Reaction Using Free-Energy Surfaces and Solvent Coordinates. *Nat. Chem.* **2013**, *5* (7), 566–571. https://doi.org/10.1038/nchem.1660.
(38) Branduardi, D.; Gervasio, F. L.; Parrinello, M. From A to B in Free Energy Space. *J. Chem. Phys.* **2007**, *126* (5), 54103–54112. https://doi.org/10.1063/1.2432340.
(39) Zinovjev, K.; Tuñón, I. Exploring Chemical Reactivity of Complex Systems with Path-Based Coordinates: Role of the Distance Metric. *J. Comput. Chem.* **2014**, *35* (23), 1672–1681. https://doi.org/10.1002/jcc.23673.
(40) Zinovjev, K.; Tuñón, I. Reaction Coordinates and Transition States in Enzymatic Catalysis. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8* (1), e1329. https://doi.org/10.1002/wcms.1329.
(41) Bonomi, M.; Branduardi, D.; Bussi, G.; Camilloni, C.; Provasi, D.; Raiteri, P.; Donadio, D.; Marinelli, F.; Pietrucci, F.; Broglia, R. A.; Parrinello, M. PLUMED: A Portable Plugin for Free-Energy Calculations with Molecular Dynamics. *Comput. Phys. Commun.* **2009**, *180* (10), 1961–1972. https://doi.org/10.1016/J.CPC.2009.05.011.
(42) Tribello, G. A.; Bonomi, M.; Branduardi, D.; Camilloni, C.; Bussi, G. PLUMED 2: New Feathers for an Old Bird. *Comput. Phys. Commun.* **2014**, *185* (2), 604–613. https://doi.org/10.1016/J.CPC.2013.09.018.
(43) Laio, A.; Parrinello, M. Escaping Free-Energy Minima. *Proc. Natl. Acad. Sci. U. S. A.* **2002**, *99* (20), 12562–12566. https://doi.org/10.1073/pnas.202427399.
(44) Bussi, G.; Branduardi, D. Free-Energy Calculations with Metadynamics: Theory and Practice. In *Reviews in Computational Chemistry Volume 28*; 2015; pp 1–49. https://doi.org/10.1002/9781118889886.ch1.
(45) Field, M. J.; Bash, P. A.; Karplus, M. A Combined Quantum Mechanical and Molecular Mechanical Potential for Molecular Dynamics Simulations. *J. Comput. Chem.* **1990**, *11* (6), 700–733. https://doi.org/10.1002/jcc.540110605.
(46) Gilbert, J. C.; Nocedal, J. Global Convergence Properties of Conjugate Gradient Methods for Optimization. *SIAM J. Optim.* **1992**, *2* (1), 21–42. https://doi.org/10.1137/0802003.
(47) Banerjee, A.; Adams, N.; Simons, J.; Shepard, R. Search for Stationary Points on Surfaces. *J. Phys. Chem.* **1985**, *89* (1), 52–57. https://doi.org/10.1021/j100247a015.
(48) Cerjan, C. J.; Miller, W. H. On Finding Transition States. *J. Chem. Phys.* **1981**, *75* (6), 2800–2806. https://doi.org/10.1063/1.442352.
(49) Simons, J.; Joergensen, P.; Taylor, H.; Ozment, J. Walking on Potential Energy Surfaces. *J. Phys. Chem.* **1983**, *87* (15), 2745–2753. https://doi.org/10.1021/j100238a013.
(50) Baker, J. An Algorithm for the Location of Transition States. *J. Comput. Chem.* **1986**, *7* (4), 385–395. https://doi.org/10.1002/jcc.540070402.
(51) Qin, A. K. K.; Huang, V. L. L.; Suganthan, P. N. N. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evol. Comput.* **2009**, *13* (2), 398–417. https://doi.org/10.1109/TEVC.2008.927706.

(52) Page, M.; McIver, J. W. On Evaluating the Reaction Path Hamiltonian. *J. Chem. Phys.* **1988**, *88* (2), 922–935. https://doi.org/10.1063/1.454172.

(53) Melissas, V. S.; Truhlar, D. G.; Garrett, B. C. Optimized Calculations of Reaction Paths and Reaction-Path Functions for Chemical Reactions. *J Chem Phys* **1992**, *96* (8), 5758–5772.

(54) Swope, W. C.; Andersen, H. C.; Berens, P. H.; Wilson, K. R. A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters. *J. Chem. Phys.* **1982**, *76* (1), 637–649. https://doi.org/10.1063/1.442716.

(55) Berendsen, H. J. C.; Postma, J. P. M.; Van Gunsteren, W. F.; Dinola, A.; Haak, J. R. Molecular Dynamics with Coupling to an External Bath. *J. Chem. Phys.* **1984**, *81* (8), 3684–3690. https://doi.org/10.1063/1.448118.

(56) Paterlini, M. G.; Ferguson, D. M. Constant Temperature Simulations Using the Langevin Equation with Velocity Verlet Integration. *Chem. Phys.* **1998**, *236* (1–3), 243–252. https://doi.org/10.1016/S0301-0104(98)00214-6.

(57) Maragliano, L.; Vanden-Eijnden, E. On-the-Fly String Method for Minimum Free Energy Paths Calculation. *Chem. Phys. Lett.* **2007**, *446* (1–3), 182–190. https://doi.org/10.1016/J.CPLETT.2007.08.017.

(58) Zinovjev, K.; Tuñón, I. Adaptive Finite Temperature String Method in Collective Variables. *J. Phys. Chem. A* **2017**, *121* (51), 9764–9772. https://doi.org/10.1021/acs.jpca.7b10842.

(59) Grote, R. F.; Hynes, J. T. The Stable States Picture of Chemical Reactions. II. Rate Constants for Condensed and Gas Phase Reaction Models. *J. Chem. Phys.* **1980**, *73* (6), 2715–2732. https://doi.org/10.1063/1.440485.

(60) Gertner, B. J.; Wilson, K. R.; Hynes, J. T. Nonequilibrium Solvation Effects on Reaction Rates for Model S $_N$ 2 Reactions in Water. *J. Chem. Phys.* **1989**, *90* (7), 3537–3558. https://doi.org/10.1063/1.455864.

(61) Roca, M.; Moliner, V.; Tuñón, I.; Hynes, J. T. Coupling between Protein and Reaction Dynamics in Enzymatic Processes: Application of Grote-Hynes Theory to Catechol O-Methyltransferase. *J. Am. Chem. Soc.* **2006**, *128* (18), 6186–6193. https://doi.org/10.1021/ja058826u.

(62) Jónsson, H.; Mills, G.; Jacobsen, K. W. Nudged Elastic Band Method for Finding Minimum Energy Paths of Transitions; World Scientific Pub Co Pte Lt, 1998; pp 385–404. https://doi.org/10.1142/9789812839664_0016.

(63) Henkelman, G.; Jónsson, H. Improved Tangent Estimate in the Nudged Elastic Band Method for Finding Minimum Energy Paths and Saddle Points. *J. Chem. Phys.* **2000**, *113* (22), 9978–9985. https://doi.org/10.1063/1.1323224.

(64) Herbol, H. C.; Stevenson, J.; Clancy, P. Computational Implementation of Nudged Elastic Band, Rigid Rotation, and Corresponding Force Optimization. *J. Chem. Theory Comput.* **2017**, *13* (7), 3250–3259. https://doi.org/10.1021/acs.jctc.7b00360.

(65) Martí, S.; Moliner, V.; Tuñón, I. Improving the QM/MM Description of Chemical Processes: A Dual Level Strategy to Explore the Potential Energy Surface in Very Large Systems. *J. Chem. Theory Comput.* **2005**, *1* (5), 1008–1016. https://doi.org/10.1021/ct0501396.

(66) Kumar, S.; Rosenberg, J. M.; Bouzida, D.; Swendsen, R. H.; Kollman, P. A. The Weighted Histogram Analysis Method for Free-Energy Calculations on Biomolecules. I. The Method. *J. Comput. Chem.* **1992**, *13* (8), 1011–1021. https://doi.org/10.1002/jcc.540130812.

(67) Souaille, M.; Roux, B. Extension to the Weighted Histogram Analysis Method: Combining Umbrella Sampling with Free Energy Calculations. *Comput. Phys. Commun.* **2001**, *135*, 40–57.

(68) Chipot, C. Free Energy Calculations in Biological Systems. How Useful Are They in Practice? *Lecture Notes in Computational Science and Engineering*. Springer Verlag 2006, pp 185–211. https://doi.org/10.1007/3-540-31618-3_12.

(69) Martí, S.; Andrés, J.; Moliner, V.; Silla, E.; Tuñón, I.; Bertrán, J. Theoretical Study of Catalytic Efficiency of a Diels–Alderase Catalytic Antibody: An Indirect Effect Produced During the Maturation Process. *Chem. - A Eur. J.* **2008**, *14* (2), 596–602. https://doi.org/10.1002/chem.200700290.

(70) Bergsma, J. P.; Gertner, B. J.; Wilson, K. R.; Hynes, J. T. Molecular Dynamics of a Model SN2 Reaction in Water. *J. Chem. Phys.* **1987**, *86* (3), 1356–1376. https://doi.org/10.1063/1.452224.

(71) Steffen, M. A Simple Method for Monotonic Interpolation in One Dimension. *Astron. Astrophys.* **1990**, *239*, 443–450.

(72) Akima, H. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. *J. ACM* **1970**, *17* (4), 589–602. https://doi.org/10.1145/321607.321609.

(73) Fritsch, F. N.; Carlson, R. E. Monotone Piecewise Cubic Interpolation. *SIAM J. Numer. Anal.* **1980**, *17* (2), 238–246. https://doi.org/10.1137/0717021.

(74) Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9* (3), 99–104. https://doi.org/10.1109/MCSE.2007.55.

(75) Arthur, D.; Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*; SODA ’07; Society for Industrial and Applied Mathematics: USA, 2007; pp 1027–1035.

(76) Kneller, G. R. Superposition Of Molecular Structures Using Quaternions. *Mol. Simul.* **1991**, *7* (1–2), 113–119. https://doi.org/10.1080/08927029108022453.

(77) Kabsch, W. A Discussion of the Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallogr. Sect. A* **1978**, *34* (5), 827–828. https://doi.org/10.1107/S0567739478001680.

(78) Kabsch, W. A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallogr. Sect. A* **1976**, *32* (5), 922–923. https://doi.org/10.1107/S0567739476001873.

(79) Farkas, Ö.; Schlegel, H. B. Methods for Optimizing Large Molecules. Part III. An Improved Algorithm for Geometry Optimization Using Direct Inversion in the Iterative Subspace (GDIIS). *Phys. Chem. Chem. Phys.* **2002**, *4* (1), 11–15. https://doi.org/10.1039/b108658h.

(80) Bennett, C. H. Efficient Estimation of Free Energy Differences from Monte Carlo Data. *J. Comput. Phys.* **1976**, *22* (2), 245–268. https://doi.org/10.1016/0021-9991(76)90078-4.

(81) Shirts, M. R.; Bair, E.; Hooker, G.; Pande, V. S. Equilibrium Free Energies from Nonequilibrium Measurements Using Maximum-Likelihood Methods. *Phys. Rev. Lett.* **2003**, *91* (14), 140601–140604. https://doi.org/10.1103/PhysRevLett.91.140601.

(82) Kästner, J.; Thiel, W. Bridging the Gap between Thermodynamic Integration and Umbrella Sampling Provides a Novel Analysis Method: "Umbrella Integration." *J. Chem. Phys.* **2005**, *123* (14), 144104–144108. https://doi.org/10.1063/1.2052648.

(83) Kästner, J.; Thiel, W. Analysis of the Statistical Error in Umbrella Sampling Simulations by Umbrella Integration. *J. Chem. Phys.* **2006**, *124* (23), 234106–234112. https://doi.org/10.1063/1.2206775.

(84) Kästner, J. Umbrella Integration in Two or More Reaction Coordinates. *J. Chem. Phys.* **2009**, *131* (3), 034109–034116. https://doi.org/10.1063/1.3175798.

(85) Balaeff, A. SODIUM - Arrange Ions around Biological Macromolecules.

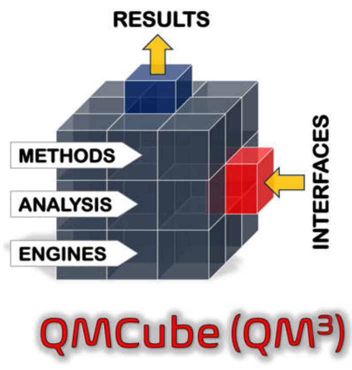
(86) Martínez, L.; Andrade, R.; Birgin, E. G.; Martínez, J. M. PACKMOL: A Package for Building Initial Configurations for Molecular Dynamics Simulations. *J. Comput. Chem.* **2009**, *30* (13), 2157–2164. https://doi.org/10.1002/jcc.21224.

(87) Pervushin, K.; Vamvaca, K.; Vögeli, B.; Hilvert, D. Structure and Dynamics of a Molten Globular Enzyme. *Nat. Struct. Mol. Biol.* **2007**, *14* (12), 1202–1206. https://doi.org/10.1038/nsmb1325.

(88) Olsson, M. H. M.; SØndergaard, C. R.; Rostkowski, M.; Jensen, J. H. PROPKA3: Consistent Treatment of Internal and Surface Residues in Empirical p K a Predictions. *J. Chem. Theory Comput.* **2011**, *7* (2), 525–537. https://doi.org/10.1021/ct100578z.

(89) MacKerell, A. D.; Bashford, D.; Bellott, M.; Dunbrack, R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-McCarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, F. T. K.; Mattos, C.; Michnick, S.; Ngo, T.; Nguyen, D. T.; Prodhom, B.; Reiher, W. E.; Roux, B.; Schlenkrich, M.; Smith, J. C.; Stote, R.; Straub, J.; Watanabe, M.; Wiórkiewicz-Kuczera, J.; Yin, D.; Karplus, M. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B* **1998**, *102* (18), 3586–3616. https://doi.org/10.1021/jp973084f.

Graphical Abstract



The QMCube suite allows to easily perform multiscale QM/MM simulations by combining highly efficient QM and MM external programs. Written in Python, it provides unified access to a wide range of cutting-edge computational methods but also supplies additional modules with extra functionalities.